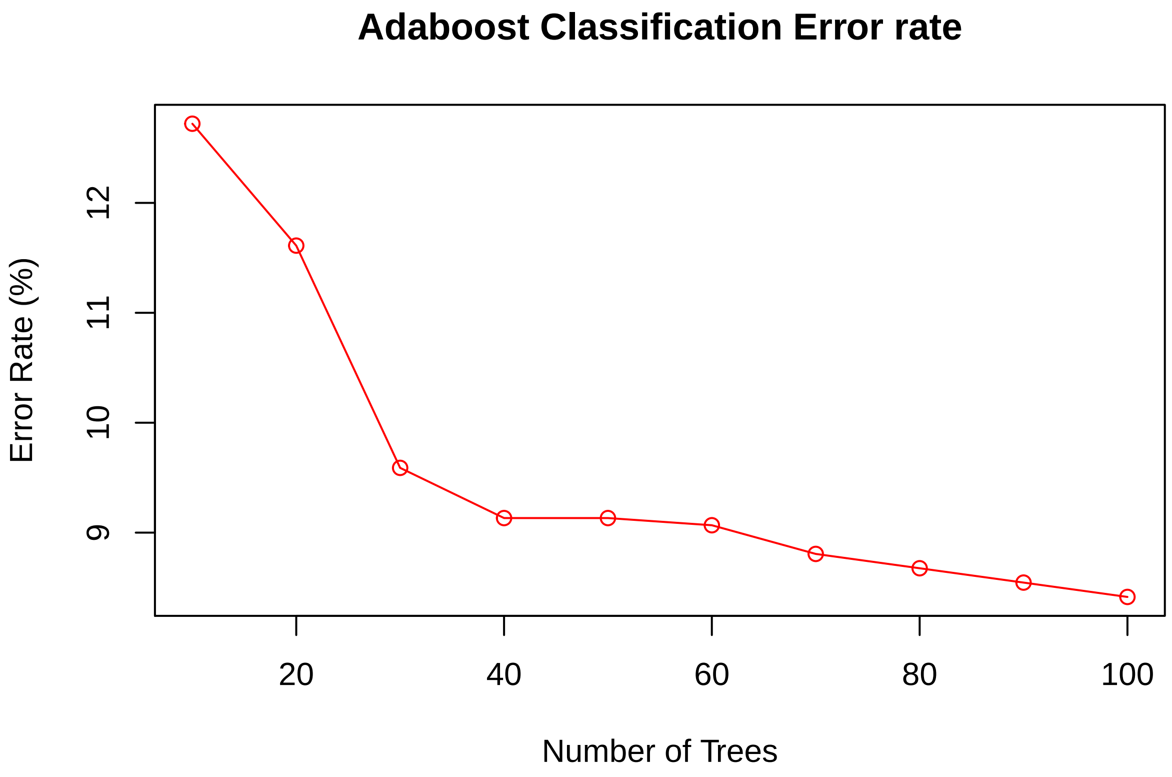
732A99 ComputerLab1 Block2

Namita Sharma

12/4/2019

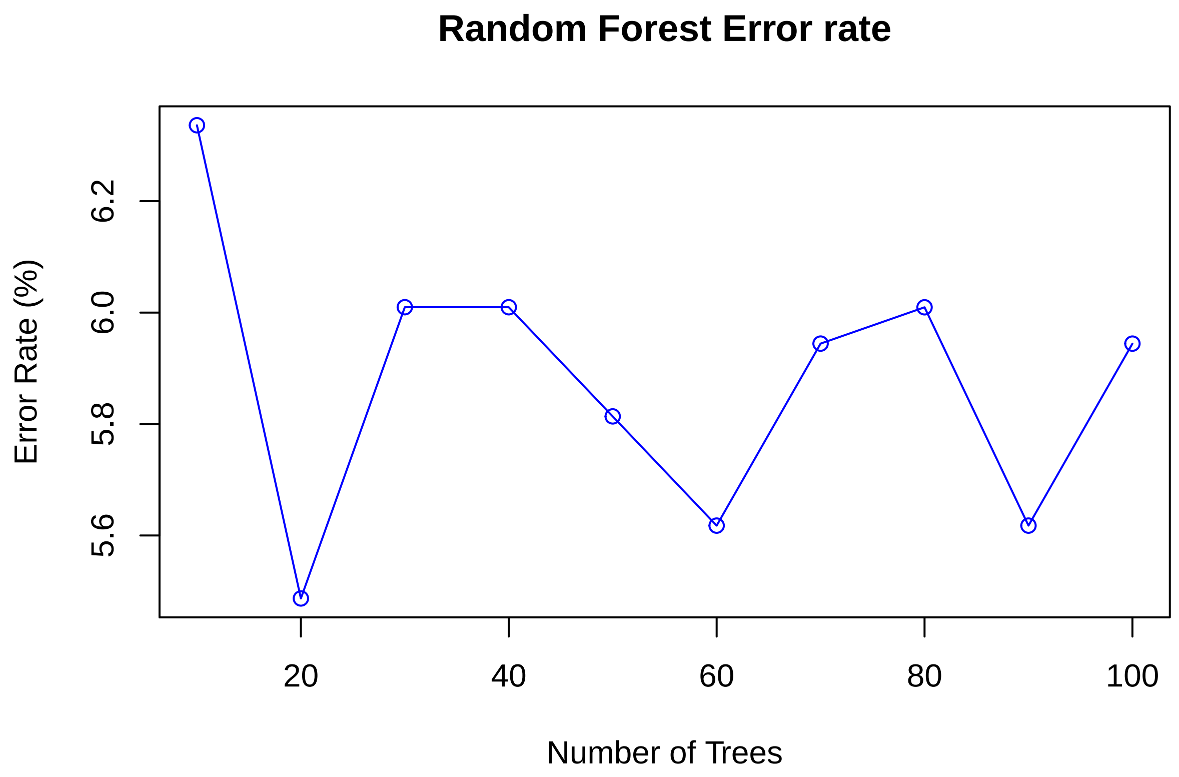
1. Ensemble Methods

1.1 Adaboost Classification Trees



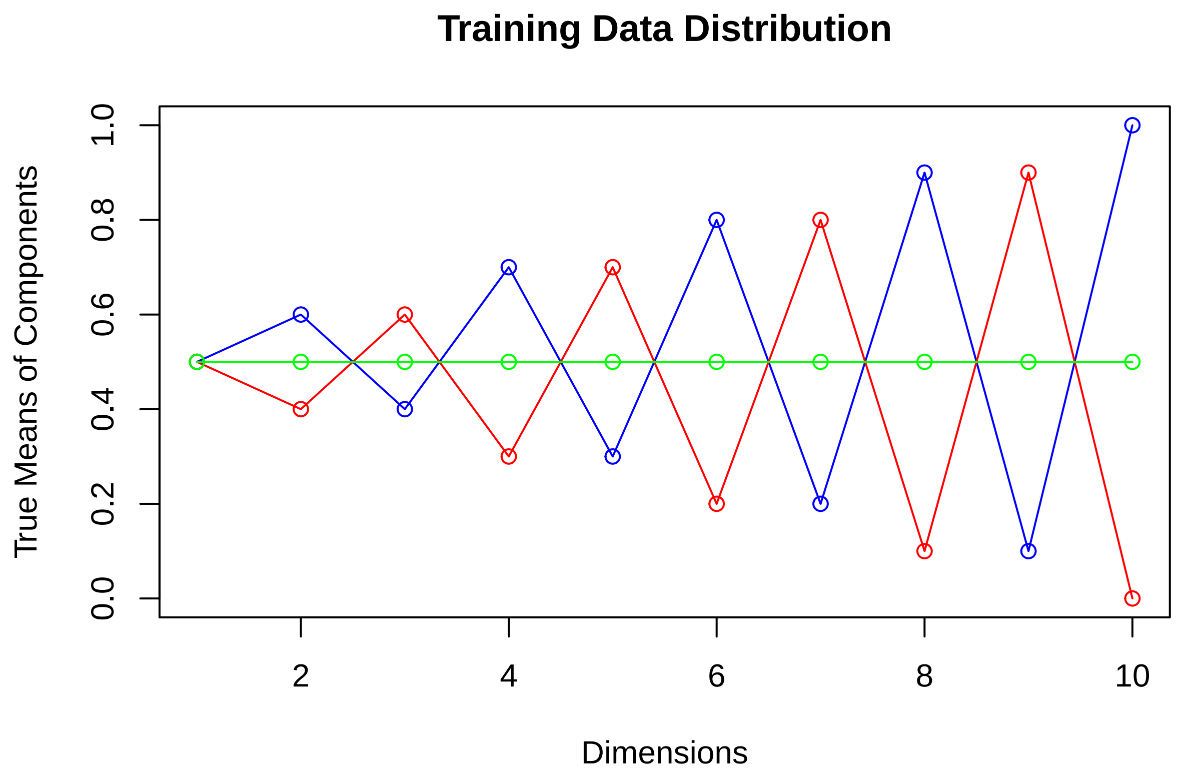
In Adaboost Classification, it is seen that the error rate reduces with an increase in the number of trees used in the algorithm. i.e. the more number of iterations for each estimate, the better the estimates on the test dataset are.

1.2 Random Forests



In Random Forests, we see that the error rate is generally much better than Adaboost classification for the same number of trees. Hence, it generalizes better than Adaboost on the test dataset. We see that error rate is minimum when number of trees used is equal to 20.

2. Mixture Models



```
## True values for Means of components
```

0.5	0.6	0.4	0.7	0.3	0.8	0.2	0.9	0.1	1.0
0.5	0.4	0.6	0.3	0.7	0.2	0.8	0.1	0.9	0.0
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

The training dataset is generated from / using the above mixture of distributions with their given mixing coefficients and component means.

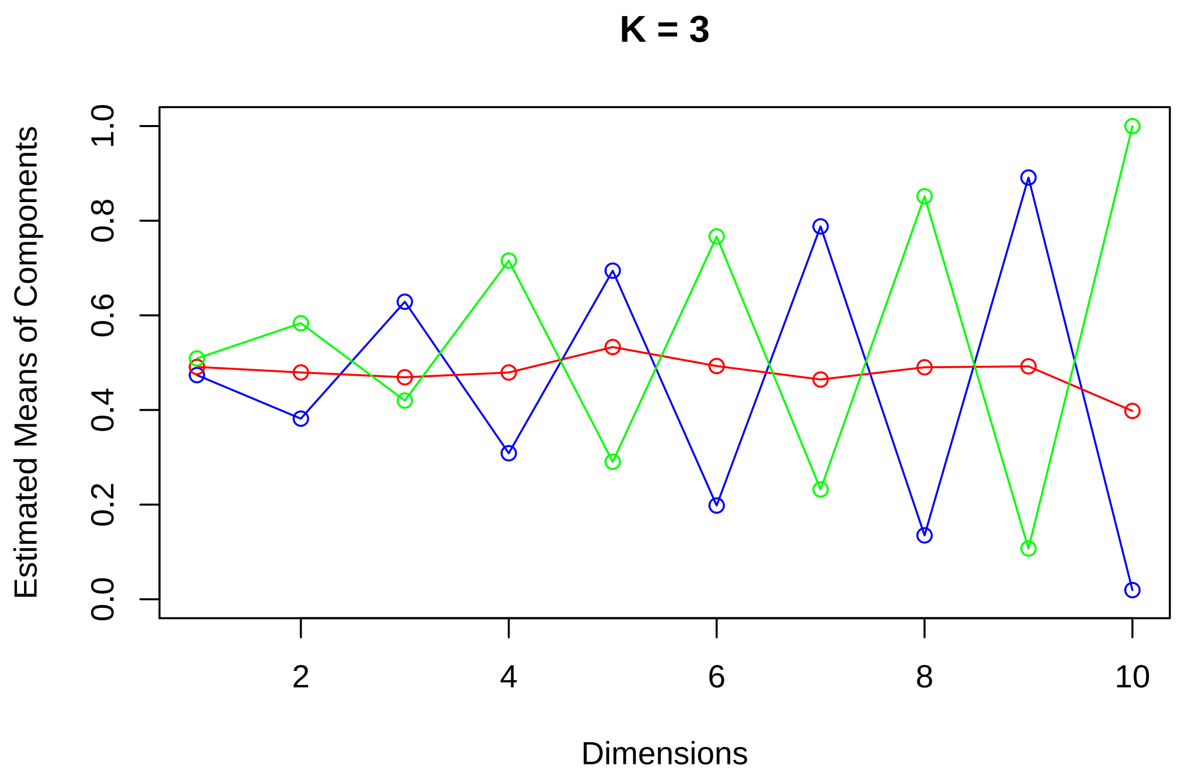
K = 3

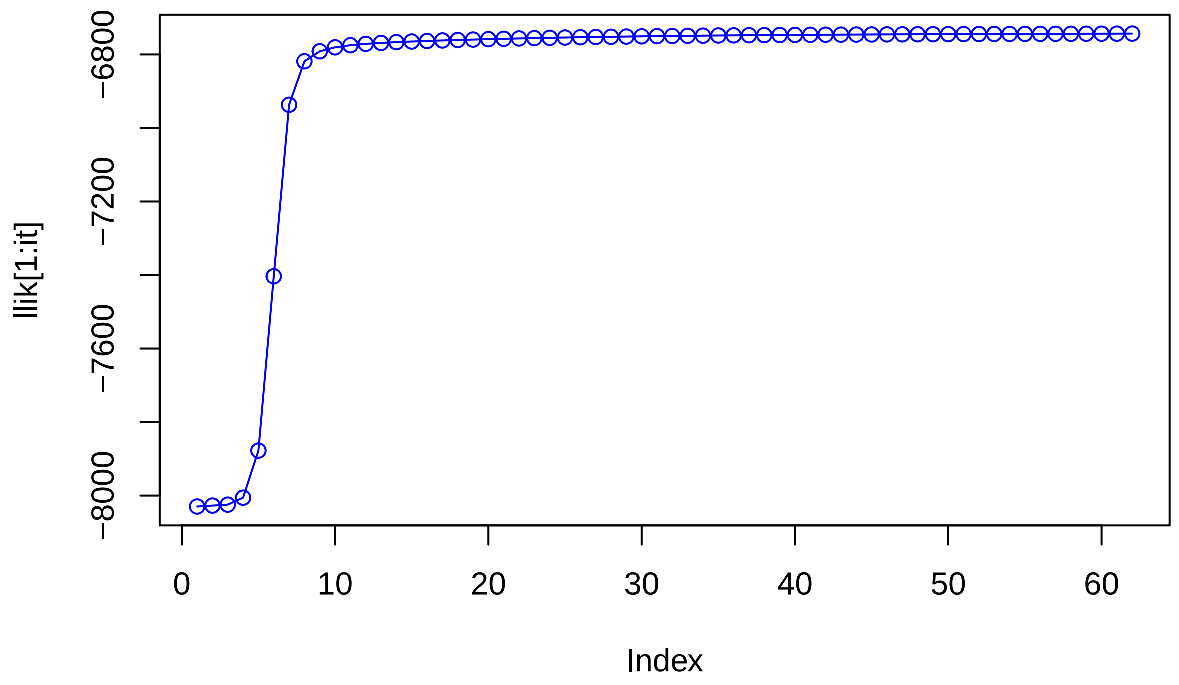
We assume the number of components in the mixture model as K = 3 and start with random values for means of those components.

```
## Initial random assignments for Means of components
```

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

```
## Final estimates for Means of components
```





We see that the EM algorithm predicts the actual distribution of the data between these components quite accurately. As seen from the graph, the final estimates of the means of the components are close to the true values of the sub population means.

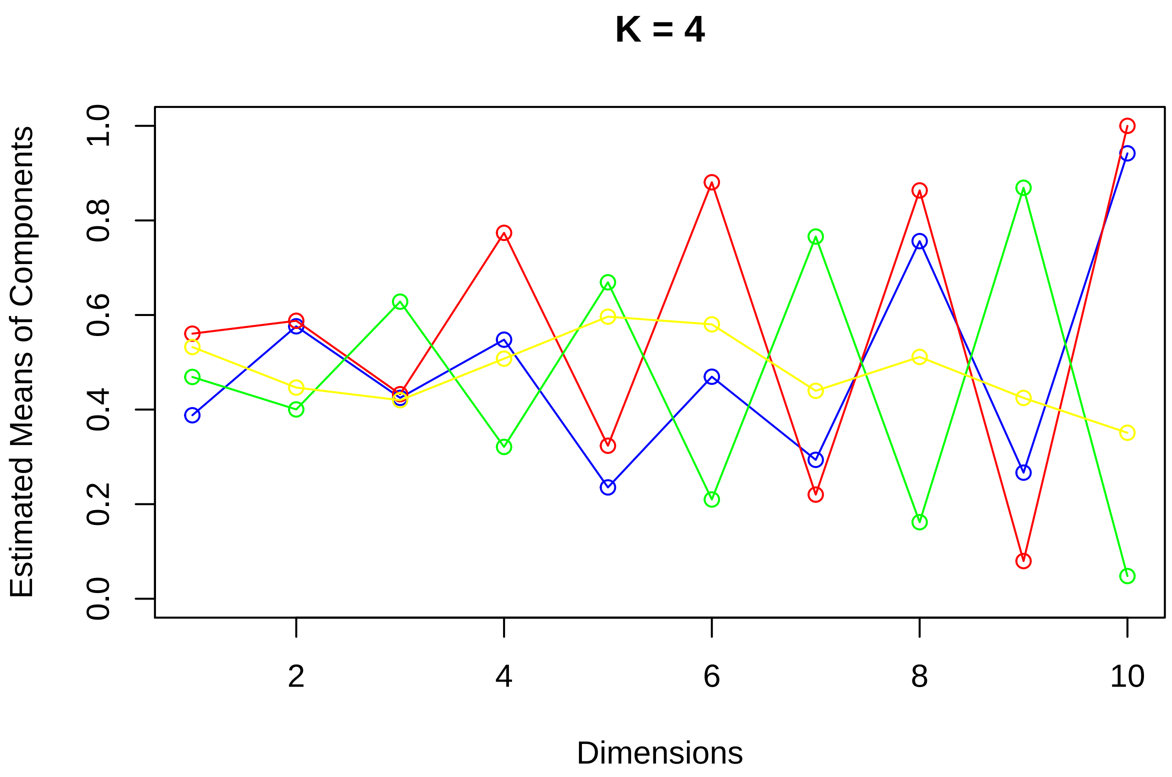
We see that as the mean values of the components converge in 97 iterations, the log likelihood values of the estimates are maximized.

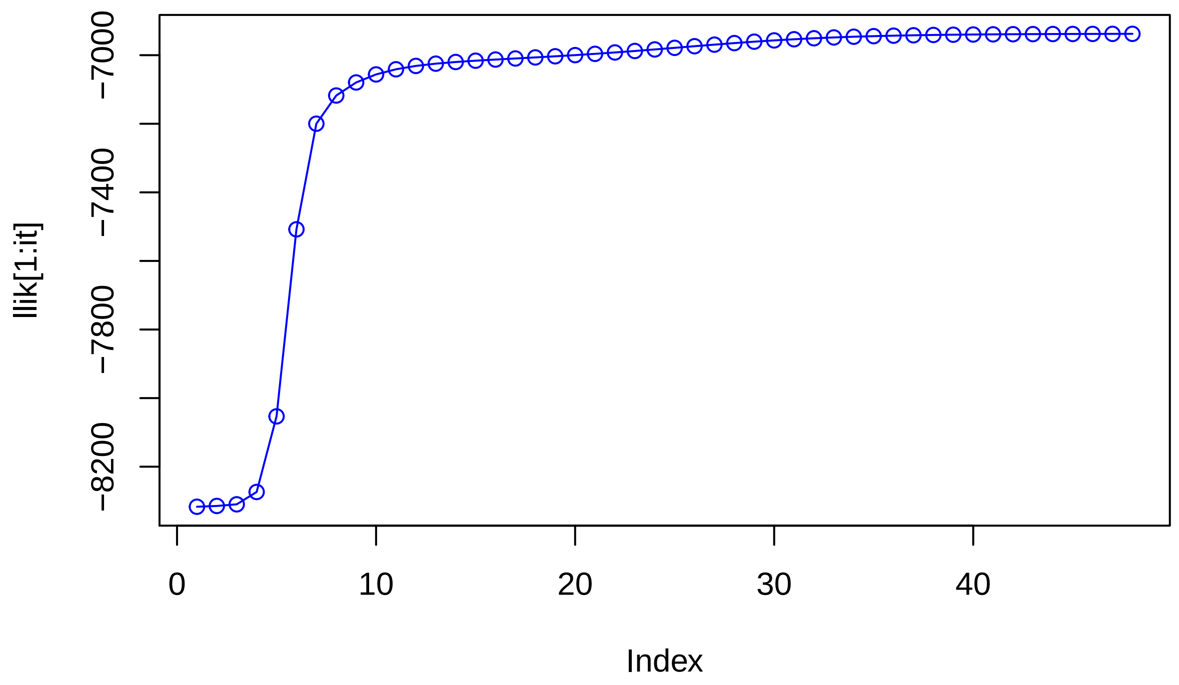
K = 4

```
## Initial random assignments for Means of components
```

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

```
## Final estimates for Means of components
```


K = 4
Estimated Means of Components
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Dimensions



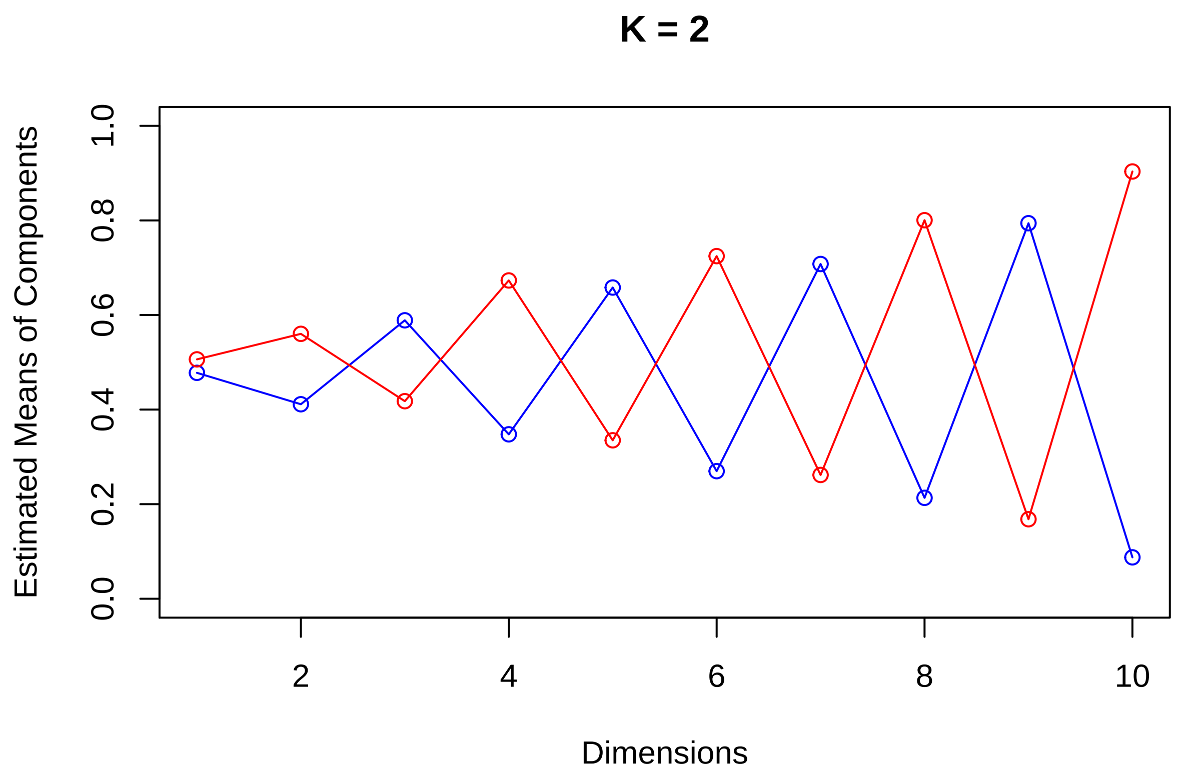
When the number of components in the mixture model is increased to K = 4, we see that the estimated mean values of the components converge in 66 iterations.

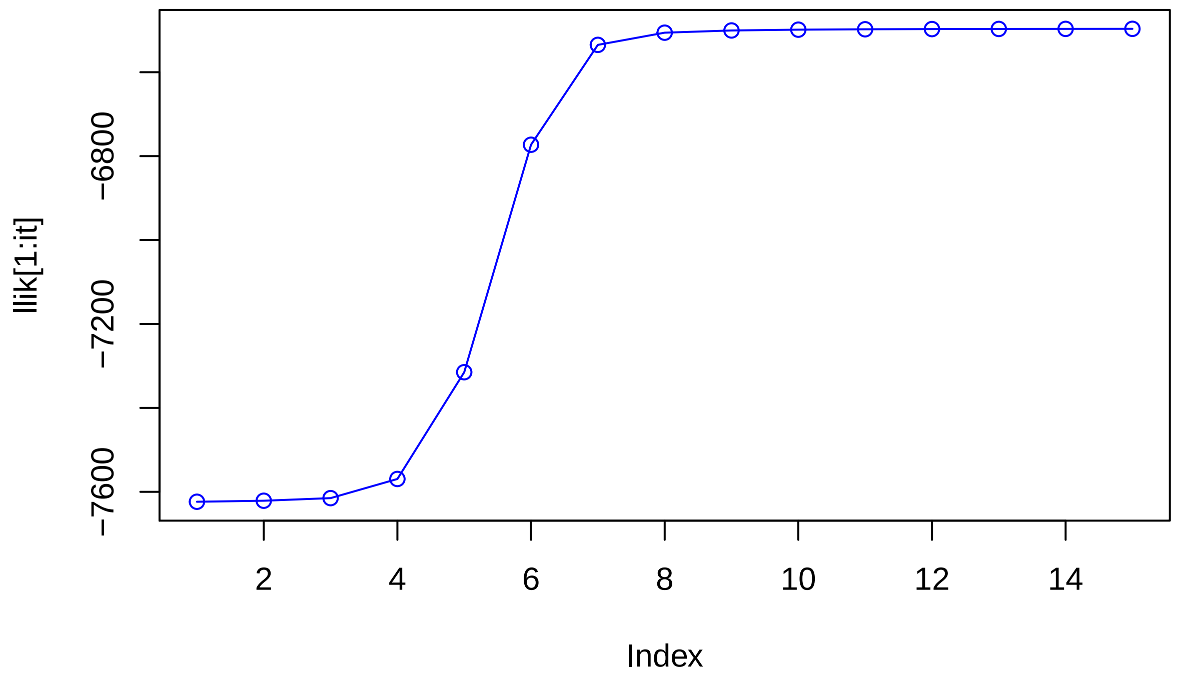
K = 2

```
## Initial random assignments for Means of components
```

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

```
## Final estimates for Means of components
```


K = 2
Estimated Means of Components
Dimensions
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10



When the number of components in the mixture model is decreased to K = 2, the estimated mean values of the components converge in just 16 iterations. And also the log likelihood values maximize at around -6600.

The model with the best fit can be chosen by evaluating and comparing their BIC scores.

Appendix

```
##############################################################################
# 1. Ensemble Methods
##############################################################################

# Import spambase data
spambase <- read.csv2(
  file   = "C:/Users/namit/Downloads/Machine Learning/Lab1 Block2/spambase.csv",
  header = TRUE)
spambase$Spam <- as.factor(spambase$Spam)

# No of observations in the dataset
n <- dim(spambase)[1]

# Divide data into training and test datasets
RNGversion("3.5.1")
set.seed(12345)
index <- sample(1:n, floor(n / 3))
```

```r
train <- spambase[-index, ]
test  <- spambase[index, ]
#-----------------------------------------------------------------------------
# Adaboost Classification Trees
library("mboost")

# No of trees in Adaboost Classification and Random Forest
trees <- seq(from = 10, to = 100, by = 10)

# Error rates for Adaboost classification Trees
mc_rate <- sapply(seq_along(trees), function(i) {
  adaboost <- blackboost(formula = Spam ~ .,                           # Adaboost model-i trees
                         data    = train,
                         family  = AdaExp(),
                         control = boost_control(mstop = trees[i]))

  pred   <- predict(object = adaboost, newdata = test, type = "class")  # Predictions on test dataset
  cf_mat <- table(predicted = pred, actual = test$Spam)                 # Confusion matrix
  mc     <- (1 - (sum(diag(cf_mat)) / sum(cf_mat))) * 100               # Mis classification rate
  return(mc)
})
# Plot error rates as a function of number of trees in Adaboost classification
plot(x    = trees,
     y    = mc_rate,
     type = "o",
     col  = "red",
     xlab = "Number of Trees",
     ylab = "Error Rate (%)",
     main = "Adaboost Classification Error rate")
#-----------------------------------------------------------------------------
# Random Forests
library("randomForest")

# Error rates for Random Forest
mc_rate <- sapply(seq_along(trees), function(i) {
  rforest <- randomForest(x     = train[, -58],             # Random Forest model-i trees
                          y     = train$Spam,
                          xtest = test[, -58],
                          ytest = test$Spam,
                          ntree = trees[i])

  cf_mat <- rforest$test$confusion                          # Confusion matrix
  mc     <- (1 - (sum(diag(cf_mat)) / sum(cf_mat))) * 100  # Mis classification rate
  return(mc)
})
# Plot error rates for different random forests
plot(x    = trees,
     y    = mc_rate,
     type = "o",
     col  = "blue",
     xlab = "Number of Trees",
     ylab = "Error Rate (%)",
     main = "Random Forest Error rate")
```

```r
################################################################################
# 2. Mixture Models
################################################################################
RNGversion('3.5.1')
set.seed(1234567890)

max_it     <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations

N = 1000 # number of training points
D = 10   # number of dimensions

x       <- matrix(nrow = N, ncol = D) # training data
true_pi <- vector(length = 3)         # true mixing coefficients
true_mu <- matrix(nrow = 3, ncol = D) # true conditional distributions

true_pi      = c(1/3, 1/3, 1/3)
true_mu[1, ] = c(0.5, 0.6, 0.4, 0.7, 0.3, 0.8, 0.2, 0.9, 0.1, 1)
true_mu[2, ] = c(0.5, 0.4, 0.6, 0.3, 0.7, 0.2, 0.8, 0.1, 0.9, 0)
true_mu[3, ] = c(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)

# Plot the means vectors of the components
plot(true_mu[1, ],
     type = "o",
     col  = "blue",
     ylim = c(0,1),
     ylab = "True Means of Components",
     xlab = "Dimensions",
     main = "Training Data Distribution")
points(true_mu[2, ], type = "o", col = "red")
points(true_mu[3, ], type = "o", col = "green")

cat("True values for Means of components")
knitr::kable(true_mu)

# Producing the training data
for(n in 1:N) {
  k <- sample(1:3, 1, prob = true_pi)
  for(d in 1:D) {
    x[n, d] <- rbinom(1, 1, true_mu[k, d])
  }
}
#-------------------------------------------------------------------------------
# EM algorithm implementation
K    <- 3                             # number of guessed components
z    <- matrix(nrow = N, ncol = K) # fractional component assignments
pi   <- vector(length = K)         # mixing coefficients
mu   <- matrix(nrow = K, ncol = D) # conditional distributions
llik <- vector(length = max_it)    # log likelihood of the EM iterations

# Random initialization of the parameters
pi <- runif(K, 0.49, 0.51)
pi <- pi / sum(pi)
```

```r
for(k in 1:K) {
  mu[k, ] <- runif(D, 0.49, 0.51)
}

cat("Initial random assignments for Means of components")
knitr::kable(round(mu, 1))

for(it in 1:max_it) {
  #---------------------------------------------------------------------------
  # E-step: Computation of the fractional component assignments

  # Calculate the likelihood estimates for each data point belonging to each components
  for (n in 1:N) {
    z[n, ] <- sapply(seq(1:K), function(k) {
      likelihood <- sapply(seq(1:D), function(d) {
        dbinom(x[n, d], 1, mu[k, d])
      })
      # Multiply the mixing coefficient for each likelihood estimate
      return(prod(likelihood) * pi[k])
    })
    # Probability that the observation belongs to each subpopulation
    z[n, ] <- z[n, ] / sum(z[n, ])
  }

  #---------------------------------------------------------------------------
  # Log likelihood computation
  for(n in 1:N) {
    for(k in 1:K) {
      llik[it] <- llik[it] +
        z[n, k] * (log(pi[k]) + sum((x[n, ] * log(mu[k, ])) + ((1 - x[n, ]) * log(1 - mu[k, ]))))
    }
  }

  # Stop if the log likelihood has not changed significantly
  if (it!= 1 && abs(llik[it - 1] - llik[it]) < min_change) {
    break
  }

  #---------------------------------------------------------------------------
  # M-step: ML parameter estimation from the data and fractional component assignments

  # ML estimate for mixing coefficients
  pi <- sapply(seq(1:K), function(k) {
    sum(z[, k]) / N
  })
  # ML estimates for component means mu
  for(k in 1:K) {
    x_weighted <- x[, 1:D] * z[, k]
    mu[k, ] <- sapply(seq(1:D), function(d) {
      sum(x_weighted[, d]) / sum(z[, k])
    })
  }
}
```

```r
# Plot the estimated mean vectors of the components
cat("Final estimates for Means of components")
plot(mu[1, ],
     type = "o",
     col  = "blue",
     ylim = c(0,1),
     ylab = "Estimated Means of Components",
     xlab = "Dimensions",
     main = paste("K =", K))
points(mu[2, ], type = "o", col = "red")
points(mu[3, ], type = "o", col = "green")

# Plot the likelihood graph
plot(llik[1:it], type = "o", col = "blue")
```